**Documentation**

Advanced Filtering AND search

**Objective:**

This project involves developing an advanced search and filtering system for tasks and boards in a task management application. The feature enables users to perform multi-criteria filtering and dynamically save filters for future use. Implement advanced search and filtering options across tasks and boards. Provide a user-friendly UI for filter input and display results efficiently. Develop a robust backend querying system for processing filter criteria.

**Technologies Used:**

* **Frontend**: React.js, CSS/Material-UI/Tailwind
* **Backend**: Node.js, Express.js, MongoDB (with Mongoose)
* **Database**: MongoDB

**Key Features**

1. **Multi-Criteria Filtering:**
   * Filters include status, assignee, priority, due date, and task name.
   * Combine multiple filters dynamically to refine results.
2. **Dynamic Saved Filters:**
   * Allow users to save filter configurations.
   * Enable retrieval of saved filters for reuse**.**
3. **Frontend Functionality:**
   * Intuitive and interactive UI resembling Flipkart’s filter design.
   * Display filtered results in real-time.
4. **Backend Functionality:**
   * Efficient querying mechanism to handle complex filter combinations.
   * Optimize database performance with indexes and query optimization.

**Implementation Details**

**Frontend**

1. **UI Components**:
   * **Filter Sidebar**:
     + Multi-select dropdowns for **status**, **assignee**, and **priority**.
     + Date pickers for filtering by due date.
     + Input field for task name search.
   * **Filter Display Area**:
     + Shows currently applied filters.
   * **Save Filter Option**:
     + Save the current filter configuration with a user-defined name.
2. **User Experience**:
   * **Dynamic updates**: Filters update results in real-time.
     + Clear and Reset Options: Allow users to clear specific filters or reset all filters.

**Backend**

1. **Filter Query System**:
   * Parse filter criteria from the frontend into a MongoDB query.
   * Use Mongoose models to fetch data efficiently.
2. **Dynamic Saved Filters**:
   * Save user-specific filter configurations in a SavedFilters collection.
   * Allow CRUD operations for saved filters.
3. **Optimizations**:
   * Index fields used for filtering, such as **status**, **assignee**, and **priority**.
   * Use aggregation pipelines for complex filtering logic.
4. **API Endpoints**:
   * **GET /tasks**: Fetch tasks based on filter criteria.
   * **POST /saved-filters**: Save a new filter configuration.
   * **GET /saved-filters**: Retrieve all saved filters for a user.
   * **DELETE /saved-filters/**

: Delete a saved filter.

**Example Use Case:**

1. **Filter Scenario**:
   * User applies filters:
     + Status: *In Progress*
     + Assignee: *John Doe*
     + Priority: *High*
     + Due Date: *2024-11-30*
2. **Saved Filter**:
   * User saves this configuration as “High Priority - John Tasks.”
3. **Revisiting Filters**:
   * User selects the saved filter to quickly apply the same criteria in the future.

**API Documentation: Task Management APIs:**

**Endpoints**

**1. Search Tasks**

GET /api/tasks/search

**Description**: Retrieve a list of tasks based on multiple filter criteria with pagination.

{

"tasks": [

{

"taskName": "Task 1",

"assigneeName": "John Doe",

"dueDate": "2024-11-20",

"status": "Open",

"priority": "High"

}

],

"total": 50,

"page": 1,

"totalPages": 5

}

**Get Task by ID**

GET /api/tasks/:id

**Description**: Retrieve details of a specific task by its unique ID.

{

"taskName": "Task 1",

"assigneeName": "John Doe",

"dueDate": "2024-11-20",

"status": "Open",

"priority": "High"

}

**3. Create Task**

POST /api/tasks

**Description**: Create a new task.

{

"\_id": "12345",

"taskName": "Task 1",

"assigneeName": "John Doe",

"dueDate": "2024-11-20",

"status": "Open",

"priority": "High"

}

**Future Enhancements:**

1. **Sorting**: Add query parameters for sorting tasks by due date, priority, etc.
2. **Bulk Operations**: Support bulk delete or update for tasks.
3. **Authorization**: Integrate user authentication to restrict access to certain tasks.

This API is designed to provide a comprehensive solution for managing tasks in a flexible and efficient manner.

**TaskSearch Component**

**Features**

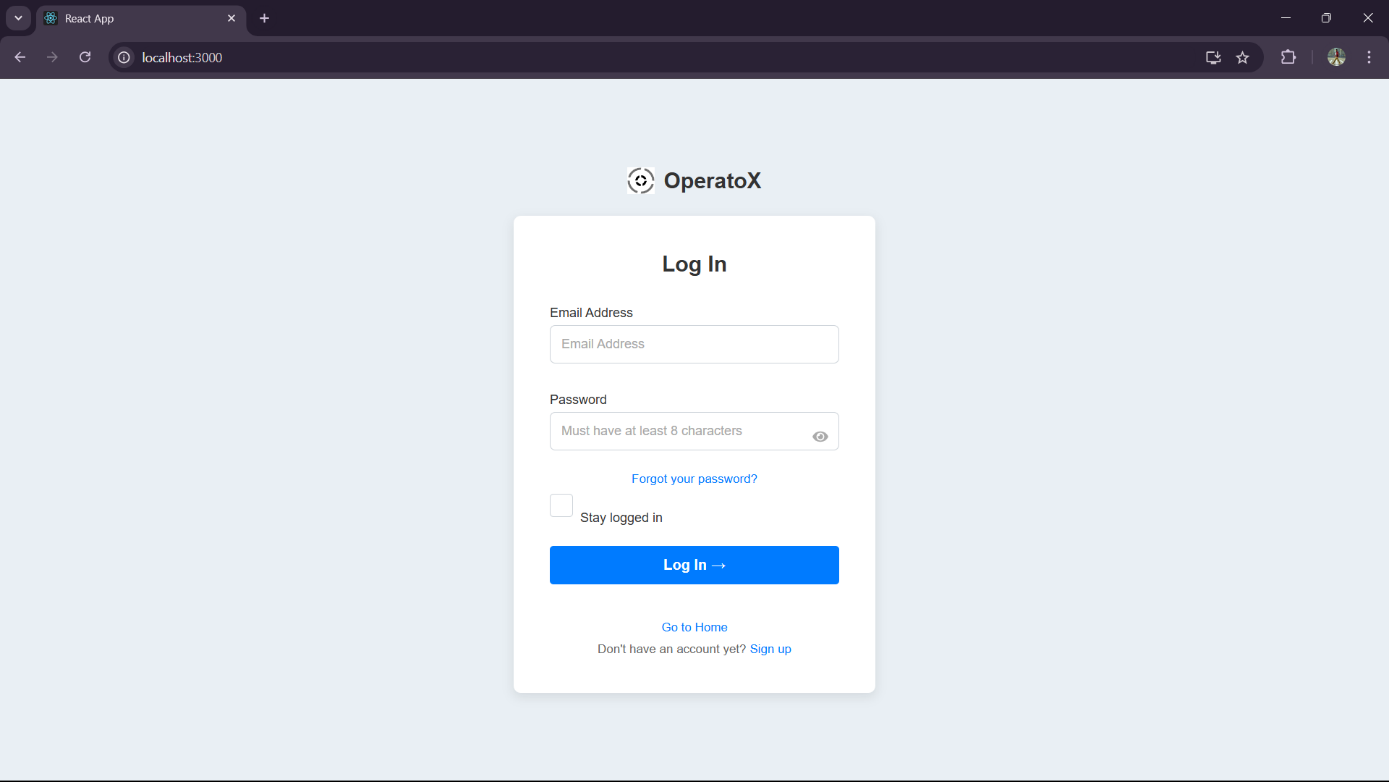
1. **Filter Form**:
   * Allows users to filter tasks by the following fields:
     + Task Name
     + Assignee Name
     + Due Date
     + Status (Pending, In Progress, Completed)
     + Priority (Low, Medium, High)
   * Search filters update dynamically upon user input.
2. **Pagination**:
   * Displays tasks in pages.
   * Controls for navigating between pages.
3. **Entries per Page**:
   * Users can control the number of tasks displayed per page (5, 10, 25, 50).
4. **Dynamic Data Fetching**:
   * Communicates with the backend API to fetch filtered and paginated task data using Axios.

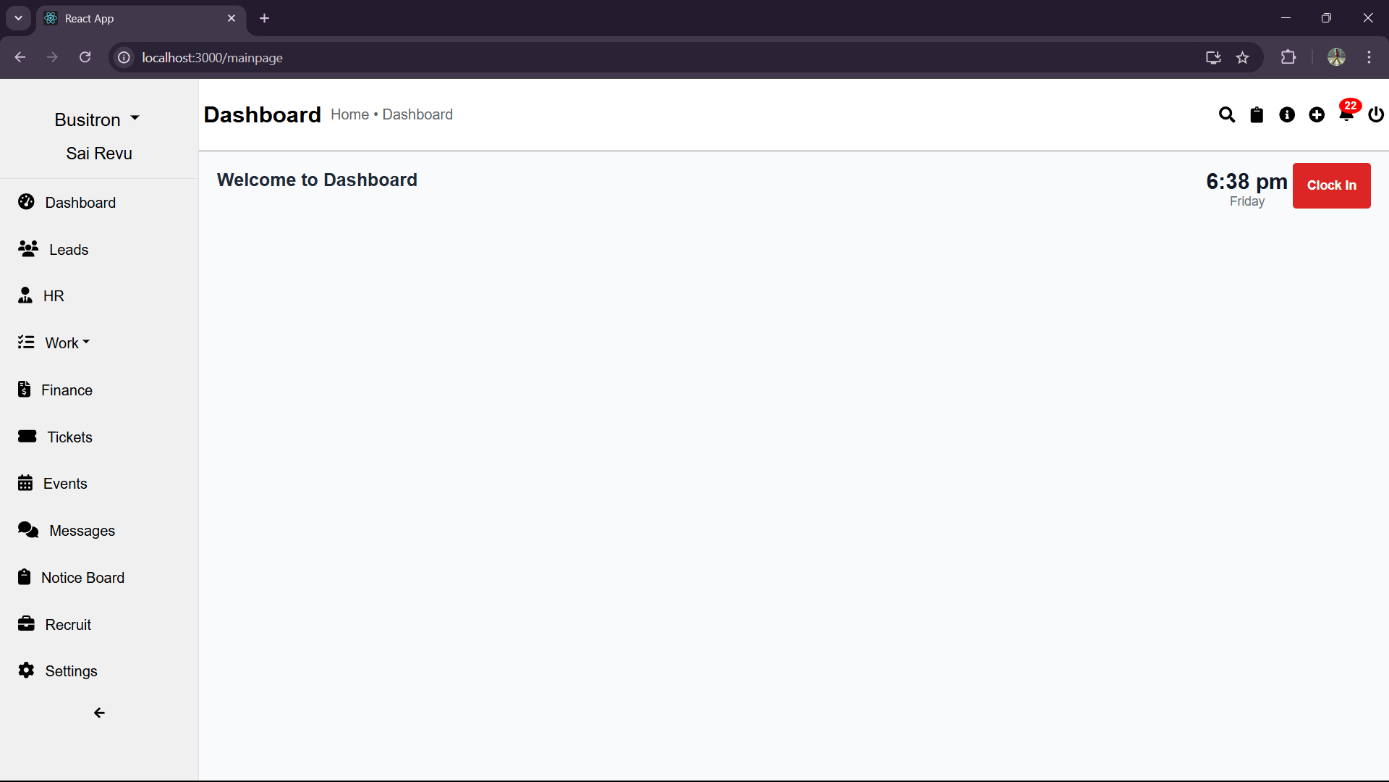
### **API Integration:**

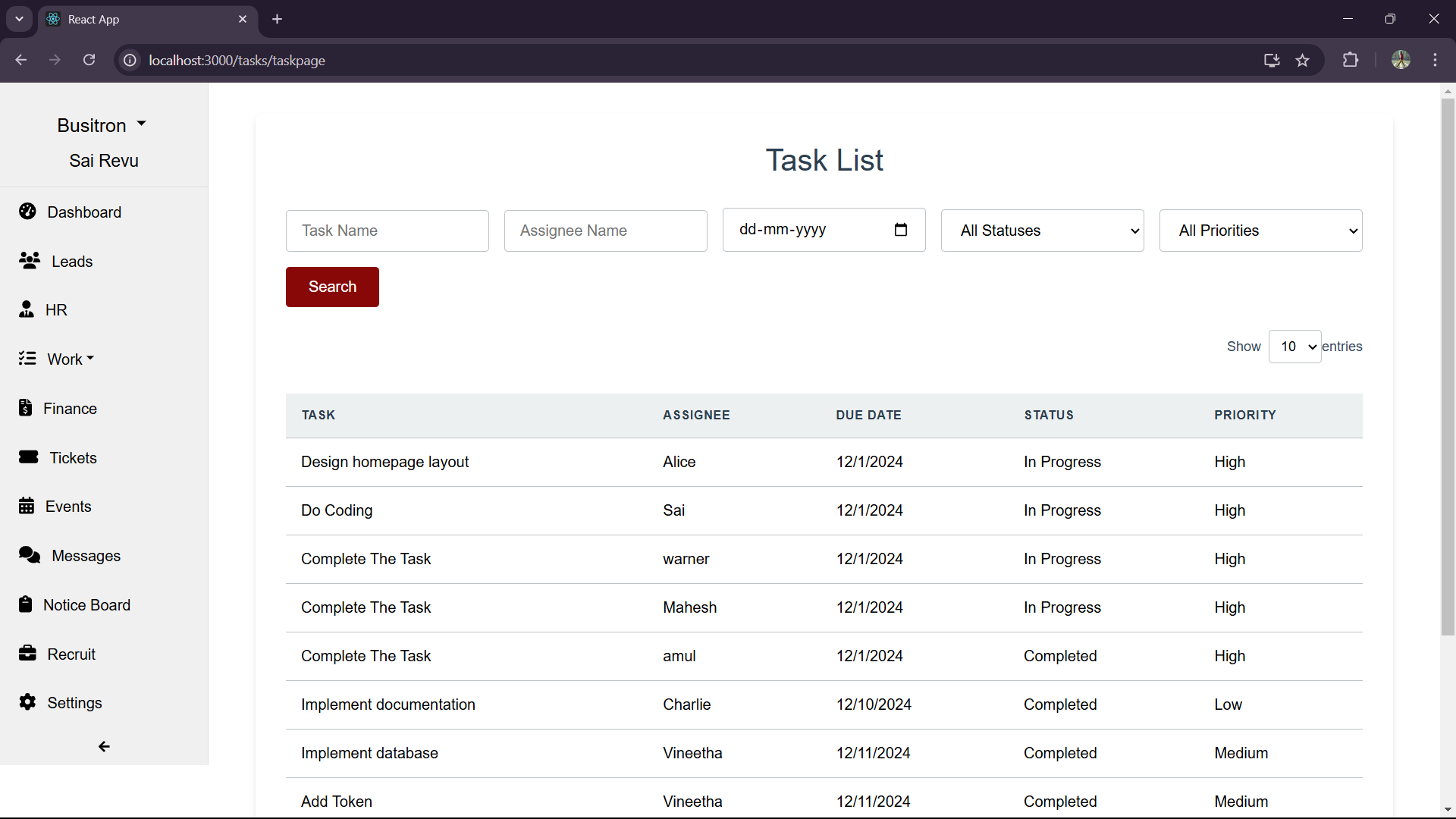
**API Endpoint**:

http://localhost:5001/api/tasks/search

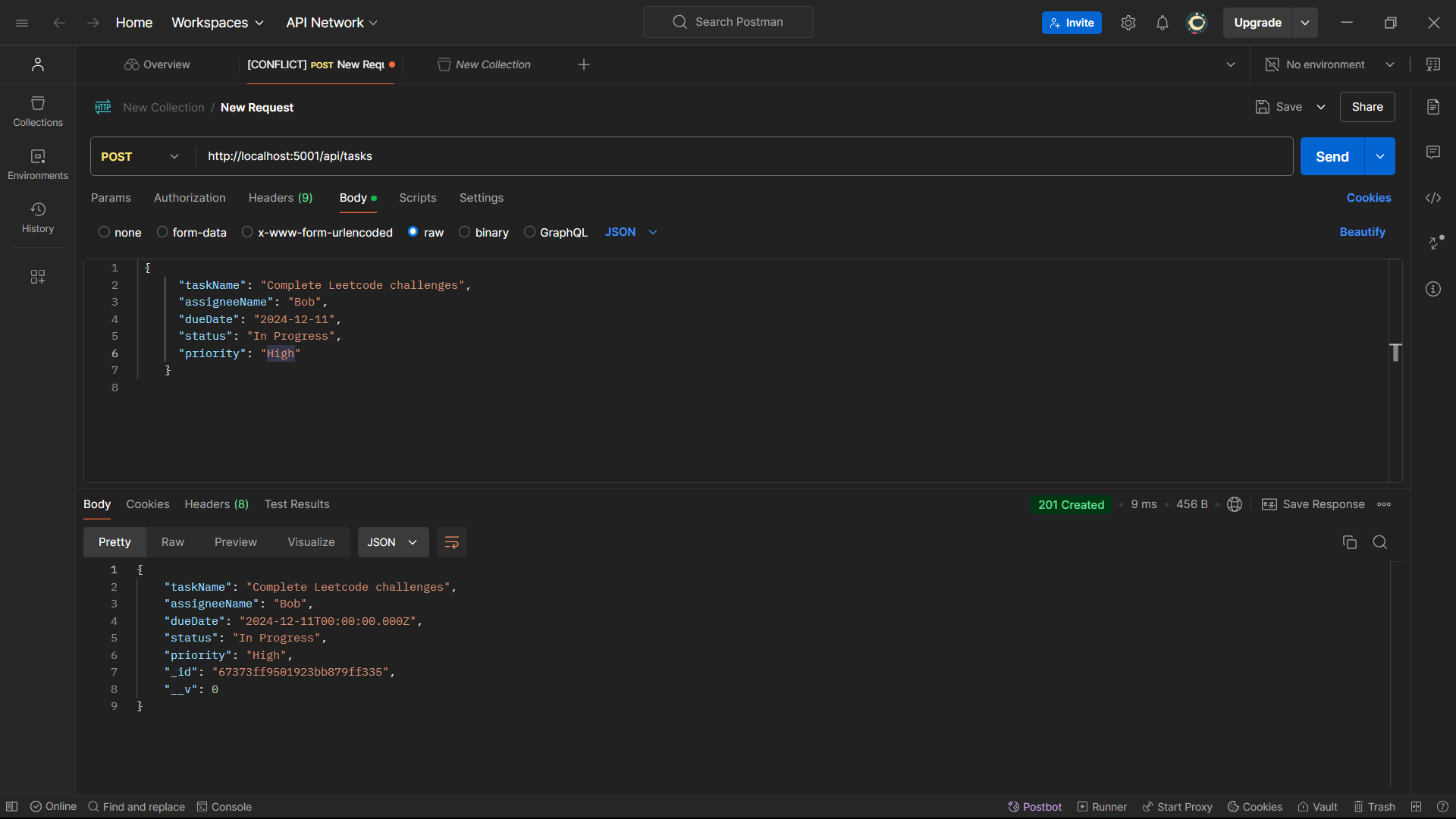
**Frontend Output**







**Backend Output:**

**

**Conclusion:**

The Advanced Filtering and Searching System was successfully developed using the MERN stack, meeting all project objectives. The frontend features a user-friendly multi-criteria filter interface and saved filter functionality. The backend ensures efficient querying for fast task retrieval. The system enhances task management by providing seamless and flexible search capabilities. Overall, the solution improves project workflow efficiency and user experience.

# Thank You